Virtual network function placement in satellite edge computing with a potential game approach

Article (Accepted Version)

Virtual Network Function Placement in Satellite Edge Computing with a Potential Game Approach

Xiangqiang Gao, Rongke Liu, *Senior Member, IEEE,* and Aryan Kaushik, *Member, IEEE*

***Abstract*—Satellite networks, as a supplement to terrestrial networks, can provide effective computing services for Internet of Things (IoT) users in remote areas. Due to the resource limitation of satellites, such as in computing, storage, and energy, a computation task from an IoT user can be divided into several parts and cooperatively accomplished by multiple satellites to improve the overall operational efficiency of satellite networks. Network function virtualization (NFV) is viewed as a new paradigm in allocating network resources on-demand. Satellite edge computing combined with the NFV technology is becoming an emerging topic. In this paper, we propose a potential game approach for virtual network function (VNF) placement in satellite edge computing. The VNF placement problem aims to minimize the deployment cost for each user request, furthermore, we consider that a satellite network should provide computing services for as many user requests as possible. We formulate the VNF placement problem as a potential game to maximize the overall network payoff and analyze the problem by a game-theoretical approach. We implement a decentralized resource allocation algorithm based on a potential game (PGRA) to tackle the VNF placement problem by finding a Nash equilibrium. Finally, we conduct the experiments to evaluate the performance of the proposed PGRA algorithm. The simulation results show that the proposed PGRA algorithm can effectively address the VNF placement problem in satellite edge computing.**

***Index Terms*—Network function virtualization (NFV), satellite edge computing, virtual network function (VNF), resource allocation, potential game.**

I. Introduction

WITH the rapid development of the Internet of Things (IoT) and edge computing technologies, IoT users can be distributed in order to provide wide coverage services in remote areas, e.g., environment monitoring, ocean transportation, smart grid, etc., [1]. Considering that IoT users have low latency requirements, limited computing capabilities and battery power, computation tasks from IoT users can be offloaded to nearby edge servers for further performing, where edge servers are usually deployed at base stations (BSs) [2]. However, terrestrial networks have not been established in some remote areas of deserts, oceans, and mountains, due to high network construction costs and specific geographical conditions [3]. Therefore, it is hard to offer data collection and computation offloading for IoT users only by terrestrial networks in these remote areas. As a supplement to terrestrial networks, low earth orbit (LEO) satellite networks, which have global seamless coverage and low transmission delay time, play an important role in satellite-based IoT and edge computing [4]–[6].

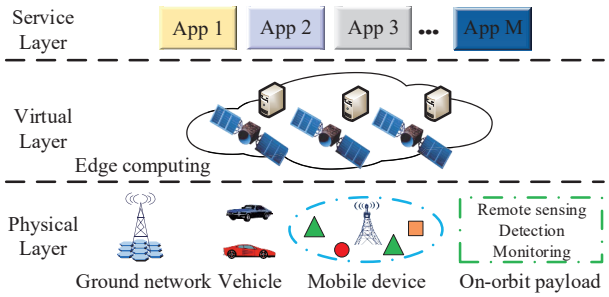


Fig. 1. Satellite edge computing framework with NFV.

For some remote areas without the coverage of terrestrial networks, LEO satellite networks can assist in gathering data from remote IoT users and transmitting them to cloud data centers on the ground for further processing [4]. Due to the nature of LEO satellite networks, the transmission delay between remote IoT users and cloud data centers will be difficult to meet the real-time requirements of IoT users. Besides, the available network bandwidths will decrease to result in the network congestion as the number of IoT users increases. Considering the service requirements of real-time processing as well as minimum network bandwidth utilization, we can deploy edge servers on LEO satellites and provide edge computing services for remote IoT users to reduce their end-to-end delay [5], such as in ocean transportation and smart grid [1]. However, LEO satellites have limited resource capacities of computing, storage, bandwidth, and energy [7]. In order to improve the operational efficiency of LEO satellite networks, multiple LEO satellites can provide computing services by the network function virtualization (NFV) technology [8] for an IoT user in a cooperative manner.

As a new paradigm in allocating network resources on-demand, NFV can support the decoupling of software and hardware equipments and enable service functions to run on commodity servers [8]. By introducing NFV to satellite edge computing, we can abstract the available resources of LEO satellite networks into a resource pool and provide agile service provisioning for IoT users on-demand [9]. Fig. 1 shows the satellite edge computing framework with NFV, which consists of physical layer, virtual layer, and service layer. Physical layer consists of remote IoT sensors, actuators,

X. Gao is with the School of Electronic and Information Engineering, Beihang University, Beijing 100191, China (e-mail: xggao@buaa.edu.cn).

R. Liu is with the School of Electronic and Information Engineering, Beihang University, Beijing 100191, China and with Shenzhen Institute of Beihang University, Shenzhen 518038, China (e-mail: rongke_liu@buaa.edu.cn).

A. Kaushik is with the School of Engineering and Informatics, University of Sussex, Brighton BN1 9RH, United Kingdom (e-mail: aryan.kaushik@sussex.ac.uk).

ground networks, etc., and can provide sensing data collection and actuator interaction. For virtual layer, the available resources of LEO satellite networks, e.g., computing, storage, bandwidth, etc., can be abstracted into a resource pool by NFV for allocating available resources to IoT users in a flexible and scalable way. Service layer is responsible for managing LEO satellite network resources and orchestrating virtual network functions (VNFs) for IoT users.

The existing work focuses on addressing the resource allocation problem for edge computing in computation offloading and VNF placement. For computation offloading, a computation task from an IoT user is considered as a whole in allocating network resources without the cooperative operation of multiple edge nodes [10], [11], where these resource allocation strategies can not fully utilize the limited resources of edge nodes. Most of the existing work about VNF placement in edge computing focuses on allocating available network resources on-demand and optimizing their objectives, e.g., link load ratio, end-to-end delay, or bandwidth cost, etc., [12]–[15], where terrestrial network topologies are considered to be unchanged and centralized resource management approaches are usually implemented to deploy VNFs to edge servers.

Note that satellite network topologies are variable over time due to the dynamic characteristics of satellites [16], which can bring us a new difficulty in deploying VNFs to satellites and managing network resources in a centralized way. Furthermore, there exists high processing delay for centralized resource management approaches in making VNF placement decision. Therefore, the VNF placement problem in satellite edge computing should be further investigated in a decentralized way of resource allocation.

In this paper, as per the above discussion, we investigate the VNF placement problem in satellite edge computing. One aim is to minimize the overall deployment cost of server energy, network bandwidth, and service delay jointly when we deploy the VNFs for each user request to satellites. The other aim is to provide computing services for as many user requests as possible in a satellite network when the minimum overall deployment cost is guaranteed. To this end, we propose a decentralized resource allocation approach for deploying VNFs to satellites [17], [18]. When an IoT user needs to offload its computation task to satellites for obtaining computing service, a user request from the IoT user will be first sent to satellite networks in order to obtain an access permission. The user request is considered as a service function chain (SFC) [9], consists of multiple VNFs in order, and carries the information concerning service type and resource requirements. We assume that there exists a user payoff when a user request is deployed to satellites, where the user payoff is non-negative and inversely proportional to the deployment cost. Then, the minimum deployment cost for a user request is equal to the maximum user payoff. Considering that the user payoff for each user request is non-negative, we aim to maximize the sum of all user payoffs. Therefore, we establish the VNF placement problem with maximum network payoff, which is the sum of all user payoffs [19].

To address the optimization problem of VNF placement, we formulate the problem as a potential game [20], which can be performed for making decisions in distributed computing as a non-cooperative game theory and widely used for handling the resource allocation problem in decentralized optimization algorithms [19], [21]. In potential game, a user request from an IoT user is considered as a player for finding a VNF placement strategy with maximum user payoff in a self-interested way and these players have potential conflicts in maximizing their payoffs [19]. The payoff for each player can be improved by competing available resources with other players and then a Nash equilibrium can be acquired in a gradual iteration [20]. Therefore, we implement a decentralized resource allocation algorithm based on a potential game, called as PGRA, to optimize the VNF placement strategy. In each iteration, we traverse all the available paths for a user request to find a feasible strategy with maximum user payoff, where the Viterbi algorithm [22] is used to address the VNF placement problem for each path. We assume that the resource allocation strategies for these players can be shared by a message synchronization mechanism. Our main contributions of this paper are summarized as follows:

- In the perspective of LEO satellite networks,we build the VNF placement problem with maximum network payoff, which is an integer non-linear programming problem. We want to minimize the overall deployment cost including energy consumption, network bandwidth, and service delay, and then provide computing services for as many user requests as possible in a satellite network.
- To address the VNF placement problem, we formulate the problem as a potential game and analyze the problem by a game-theoretical approach. We implement a decentralized resource allocation algorithm based on a potential game to obtain an approximate strategy profile by finding a Nash equilibrium, where the Viterbi algorithm is used to deploy the VNFs for each user request.
- We conduct the experiments to simulate and evaluate the performance of the proposed PGRA algorithm in LEO satellite networks. The simulation results show that the proposed PGRA algorithm outperforms two existing baseline algorithms of Greedy and Viterbi.

The remainder of this paper is organized as follows. Section II briefly reviews related work about resource allocation in satellite edge computing and decentralized algorithms. Section III introduces the system model of VNF placement in satellite edge computing. In Section IV, we model the problem of VNF placement with maximum network payoff and prove it to be NP-hard. The VNF placement problem is formulated as a potential game and a decentralized resource allocation algorithm is implemented for tackling the problem in Section V. Section VI discusses the performance of the proposed PGRA algorithm in LEO satellite networks. Finally, we provide the conclusion of this paper in Section VII.

II. Related Work

In this section, we first discuss the existing related work in edge computing, which includes computation offloading and VNF placement. Then we summarize the difference between the existing work and our proposed work.

TABLE I
LITERATURE REVIEW AND COMPARISON WITH THE PROPOSED WORK

Reference	Computation Offloading / VNF Placement	Objective Function	Centralized / Decentralized	Optimization Approach
[19]	Computation offloading	System cost, Number of allocated users	Decentralized	EUAGame
[23], [24]	Computation offloading	Delay, Energy consumption	Decentralized	Game theory
[17]	VNF placement	Deployment cost	Centralized	Time-slot decoupled algorithm
[12]	VNF placement	Link load ratio	Centralized	Randomized rounding approximation
[13]	VNF placement	Admission cost, Throughput	Centralized	Heuristic algorithms
[14]	VNF placement	End-to-end delay	Centralized	Graph partitioning algorithm
[15]	VNF placement	Bandwidth consumption	Centralized	Greedy algorithm
Proposed work	VNF placement	Deployment cost, Number of allocated users	Decentralized	Game theory, Viterbi algorithm

A. Computation Offloading in Edge Computing

In satellite edge computing, most of the existing work focuses on offloading computation tasks from IoT devices to satellite edge nodes [10], [25], [26]. In [10], considering traditional satellites as space edge computing nodes, the authors presented an approach of satellite edge computing to share on-board resources for IoT devices and provide computing services combined with the cloud. The on-board edge computing for nano-satellite constellations was discussed by formation flying in [25]. A fine-grained resource management in satellite edge computing was presented by the advanced K-means algorithm in [26].

As the number of IoT devices increases, decentralized mechanisms of network resource management have been a research topic in distributed networks, where potential game is widely used to address the problem of resource allocation in distributed computing [19], [23], [24]. The authors in [23] proposed a game-theoretical algorithm to minimize energy consumption and latency of users in mobile edge computing. In [19], a cost-effective edge user allocation (EUA) problem in edge computing was presented to maximize the number of served users with minimum system cost and the authors designed a decentralized algorithm by a potential game to address the EUA problem. Furthermore, computation offloading in satellite edge computing was discussed by a game-theoretical approach in [24]. However, this existing literature just considers a computation task as a whole to allocate the network resources.

B. VNF Placement in Edge Computing

For software defined satellite networks, the VNF placement problem has been investigated in [9], [17], [18]. The authors in [9] formulated the VNF placement problem as an integer non-linear programming problem in space-air ground integrated networks and proposed a greedy algorithm to address it. The authors in [17] discussed the problem of VNF placement to minimize the cost in software defined satellite networks and proposed a time-slot decoupled heuristic algorithm to solve this problem. In [18], an approach of deploying VNFs in satellite networks was presented to minimize the end-to-end service delay.

There is also the existing work concerning VNF placement in terrestrial edge computing [12]–[15]. The authors in [12] discussed the delay-aware VNF placement problem to minimize the maximum link load ratio in edge cloud computing and proposed a randomized rounding approximation algorithm to tackle the problem. In [13], the authors presented a VNF-based service provisioning by formulating the cost minimization problem and the throughput maximization problem in mobile edge computing, respectively. The authors in [14] studied the VNF resource allocation based on context-aware grouping to optimize the end-to-end delay in 5G edge networks and proposed a graph partitioning algorithm to solve it. Moreover, the authors in [15] proposed an NFV-based service framework with the aim of minimizing the total bandwidth consumption in edge computing and implemented a greedy algorithm to address it. Note that most of the existing work focuses on addressing the VNF placement problems from the perspective of users or service networks in edge computing, where these network topologies are unchanged. A few existing work related to resource management in NFV-enabled satellite networks discussed how to deploy VNFs to satellite edge servers efficiently via centralized approaches [17], [18].

Compared with the existing related work in edge computing, we build LEO satellite networks with dynamic characteristics in a time-evolving way [16], which can be variable over different time slots. From the perspective of both users and satellite networks, we propose the VNF placement problem to optimize the deployment costs of energy consumption of edge servers, bandwidth cost of satellite networks, and service delay of user requests jointly, and provide computing services for as many user requests as possible. Then we formulate the VNF placement problem as the problem of maximizing network payoff. Considering the difficulty of centralized resource management approaches in satellite networks, we analyze the characteristics of a non-cooperative potential game, which is widely used to solve the problem of resource allocation in a decentralized way, and then implement the resource allocation algorithm based on a potential game in decentralized satellite networks. Table I summarizes reviewed related work in edge computing and provides difference comparison with our proposed work.

III. SYSTEM MODEL

In this section, we discuss the system model of VNF placement in satellite edge computing [3], [27], including satellite network, user requests, and the VNF placement problem.

A. Satellite Network

We denote a satellite network as a directed graph $G(V, E)$. The parameter V indicates the set of satellite nodes, where

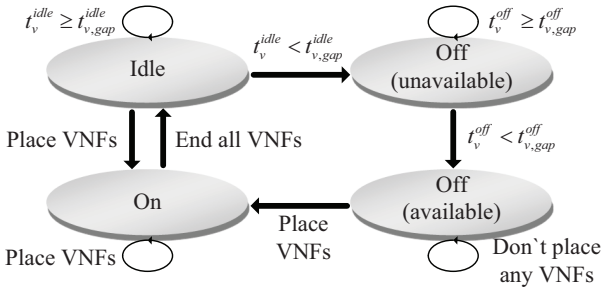


Fig. 2. State transition diagram of an edge server.

the number of satellite nodes is N. We assume that the set of resource types supported by each satellite is denoted by R and each satellite v, $v \in V$, has limited resource capacities, where two resource types of central processing unit (CPU) and storage are considered in this paper. Let us denote the r-th resource capacity of satellite v by C_v^r, $r \in R$. The parameter E indicates the set of links between satellites. We assume that each satellite has four inter-satellite links (ISLs) with neighbouring satellites, which consist of two intra-plane ISLs and two inter-plane ISLs [28], [29]. For link e, $e \in E$, we denote the bandwidth capacity by B_e and the transmission delay time by t_e. The parameter $L_{v_1}^{v_2}$ indicates the set of the d shortest paths between satellites v_1 and v_2.

Due to the limited power of satellites, one of our aims is to minimize the overall energy cost of a satellite network. We introduce a power consumption model for edge servers on satellite nodes [30]. An edge server can be considered in four states of *on*, *idle*, *unavailable off*, and *available off*. In an *on* state, an edge server can provide computing services for IoT users and will consume energy. We denote P_v^{on} as the average power consumption of an edge server on satellite v in an *on* state. If an edge server on satellite v does not provide computing services for any IoT users the edge server is considered into an *idle* state and we use P_v^{idle} to indicate the average idle power consumption. When the idle time for an edge server on satellite v is over the maximum idle threshold t_v^{idle} the edge server will be into an *unavailable off* state. If an edge server is in an *unavailable off* state, it can not provide computing services for IoT users in the next time slot. When the off time for an edge server on satellite v is greater than the minimum off threshold t_v^{off} the edge server can be in an *available off* state, that is, the edge server can provide computing services for IoT users in the next time slot. If an edge server in an *available off* state needs to provide computing services for IoT users there will exist a setup procedure for the edge server, where the state of the edge server will be converted from *off* to *on*. We assume that the period of the setup procedure is 1 time slot and the average setup power consumption is considered as the maximum power P_v^{max} of an edge server on satellite v. For *available* and *unavailable* states, the power consumption of an edge server is zero. Note that a satellite node can be considered as a router for routing traffic flows when its edge server is in an *off* state. Based on the above discussion, the state transition diagram of an edge server is shown in Fig. 2.

B. User Requests

In this paper, we consider IoT users to be on the remote areas without the coverage of terrestrial networks. When an IoT user needs to offload its computation task to satellites, a user request will be first sent to the LEO satellite network for obtaining an access permission, where the user request consists of multiple VNFs in specific order and can be considered as an SFC. We denote a set of user requests by U with M user requests, where user request u, $u \in U$, is defined as a directed graph $G(F_u, H_u)$. The set $F_u = \{f_{u,1} = s_u, f_{u,2}, \cdots, f_{u,|F_u|} = d_u\}$ indicates the set of VNFs for user request u, where $f_{u,i}$ indicates the i-th VNF of user request u, while s_u and d_u indicate the source and the destination, respectively. In satellite edge computing, we assume that the results of computation tasks processed by satellite nodes can be sent back to IoT users or transmitted to cloud data centers on ground. In these scenarios, the source and the destination of a user request can either be the same node or not. The resource requirements of each VNF include computing, storage, and execution time. We use $c_{u,i}^r$ to indicate the r-th resource requirement of $f_{u,i}$ and $t_{u,i}$ to represent the execution time of $f_{u,i}$. The set H_u describes the set of edges for user request u. An edge between f_{u,i_1} and f_{u,i_2} is denoted by $h_u^{i_1,i_2}$ and the bandwidth requirement of edge $h_u^{i_1,i_2}$ is denoted by $b_u^{i_1,i_2}$ accordingly. We define the maximum acceptable delay time for user request u as t_u^{delay}.

C. VNF Placement Problem

In this paper, we discuss the problem of dynamically deploying VNFs over varying time slots, where our main work focuses on resource allocation for deploying VNFs to provide computing services for IoT users in LEO satellite networks. Furthermore, we consider that the transmission delay between an IoT user and its access satellite is relative fixed, we also suppose that the wireless communication environment can meet the requirements of offloading computation tasks to satellites for IoT users. In that case, we can address on how to allocate available resources in LEO satellite networks to reduce the deployment costs, e.g., service delay, energy consumption, and bandwidth cost. A batch processing mode is simply used for allocating available resources of a satellite network to user requests. We assume that a batch of user requests arrive concurrently at the beginning of each time slot and they can make decisions for deploying the VNFs to satellite nodes by their optimization strategies, where user requests have different resource requirements and maximum acceptable delay time, however, the source and the destination for each user request can be known. Similar to the existing work in edge computing [19], [31], [32] and satellite networks [17], [18], we consider quasi-static scenarios for satellite networks and user requests in solving the VNF placement problem. The satellite network topology is unchanged in a time slot and can be varying with different time slots. During the computing service periods, we assume that user requests are unchanged and their service requirements can be guaranteed. For deploying VNFs in dynamic environment, we first abstract all available resources of a satellite network into a resource

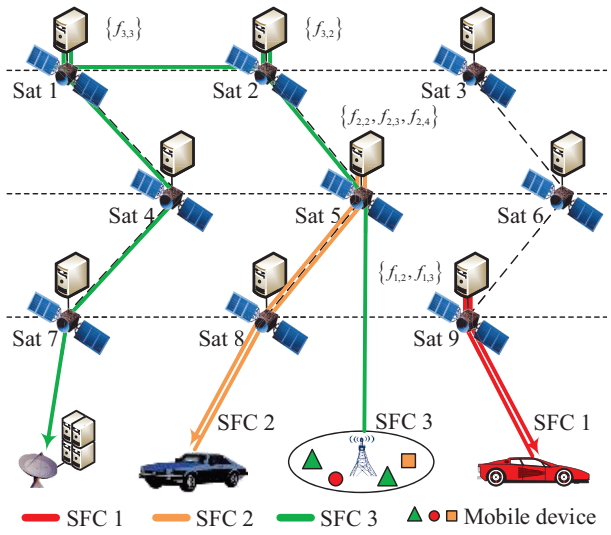


Fig. 3. Example of placing VNFs for three IoT users.

TABLE II
VNF PLACEMENT STRATEGIES FOR THREE IOT USERS.

User	Strategy
SFC1	s_1(Sat9)→ $f_{1,2}$(Sat9) → $f_{1,3}$(Sat9)→ d_1(Sat9)
SFC2	s_2(Sat8) → $f_{2,2}$(Sat5) → $f_{2,3}$(Sat5) → $f_{2,4}$(Sat5) → d_2(Sat8)
SFC3	s_3(Sat5) → $f_{3,2}$(Sat2) → $f_{3,3}$(Sat1) → Sat4 → d_3(Sat7)

pool. Before performing resource allocation algorithms for the VNF placement, we need to free the resources that are used by the completed user requests in the previous time slot into the resource pool as available resources for new user requests in the current time slot. Then under remaining available resource and service requirement constraints, we can run resource allocation algorithms to orchestrate VNFs for new user requests with maximum network payoff.

For a user request, the computation task can be offloaded to satellites by different resource allocation strategies. However, the VNF placement strategy has an impact on the network payoff. Fig. 3 shows an example of placing VNFs for three IoT users. There are 9 satellite nodes, which are represented by $\{Sat1, Sat2, \cdots, Sat9\}$, and each satellite node deploys an edge server. The number of inter-satellite links for a satellite is 4. The user requests are composed of multiple specific VNFs in sequence. The three user requests can be described by $SFC1 = \{s_1, f_{1,2}, f_{1,3}, d_1\}$, $SFC2 = \{s_2, f_{2,2}, f_{2,3}, f_{2,4}, d_2\}$, and $SFC3 = \{s_3, f_{3,2}, f_{3,3}, d_3\}$, respectively. We assume that source s_1 and destination d_1 for $SFC1$ are on the same satellite node $Sat9$, source s_2 and destination d_2 for $SFC2$ are also on the same satellite node $Sat8$. In addition, source s_3 and destination d_3 for $SFC3$ are on satellite nodes $Sat5$ and $Sat7$, respectively. We deploy VNFs $f_{1,2}$ and $f_{1,3}$ for $SFC1$ to satellite node $Sat9$, VNFs $f_{2,2}, f_{2,3}$, and $f_{2,4}$ for $SFC2$ to satellite node $Sat5$. For $SFC3$, we deploy VNF $f_{3,2}$ to satellite node $Sat2$ and VNF $f_{3,3}$ to satellite node $Sat1$, respectively. For $SFC1$, all VNFs are deployed on satellite node $Sat9$. The routing path for $SFC2$ is $\{Sat8, Sat5, Sat8\}$ and for $SFC3$ is $\{Sat5, Sat2, Sat1, Sat4, Sat7\}$. The strategies for the three IoT users are shown in Table II.

IV. PROBLEM FORMULATION

In this section, the problem of VNF placement is proposed by a mathematical method in satellite edge computing and proved to be NP-hard.

A. Problem Description

In the perspective of a satellite network, we formulate the VNF placement problem as a constrained optimization problem with maximum network payoff in satellite edge computing, where the VNF placement problem is viewed as an integer non-linear programming problem. To better discuss the problem of VNF placement, we list the main symbols for our problem description in Table III.

Let us denote a binary decision variable $x_{u,i}^v = \{0, 1\}$ to indicate whether VNF $f_{u,i}$ is placed on satellite v, where $x_{u,i}^v = 1$ if VNF $f_{u,i}$ is placed on satellite v, otherwise $x_{u,i}^v = 0$.

Another binary decision variable $y_{u,l}^{i_1,i_2} = \{0, 1\}$ is defined to describe which path is used by edge $h_u^{i_1,i_2}$. If path l is used by $h_u^{i_1,i_2}$, then $y_{u,l}^{i_1,i_2} = 1$, otherwise $y_{u,l}^{i_1,i_2} = 0$.

We also denote a binary variable $q_e^l = \{0, 1\}$ to indicate whether link e is used by path l. $q_e^l = 1$ if link e is used by path l, otherwise $q_e^l = 0$.

A binary variable $z_u = \{0, 1\}$ is used to indicate whether user request u is deployed to satellite nodes. $z_u = 1$ if user request u is deployed to satellite nodes, otherwise $z_u = 0$.

When we deploy the VNFs for a user request to satellites, the deployment cost can be composed of energy consumption, bandwidth, and service delay costs, where the three costs are normalized values.

For user request u, the used bandwidth resources u^{bw} can be described as:

$$u^{bw} = \sum_{h_u^{i_1,i_2} \in H_u} \sum_{v_1 \in V} \sum_{\substack{v_2 \in V \\ v_1 \neq v_2}} \sum_{l \in L_{v_1}^{v_2}} \sum_{e \in l} x_{u,i_1}^{v_1} \cdot x_{u,i_2}^{v_2} \cdot y_{u,l}^{i_1,i_2} \cdot q_e^l \cdot b_u^{i_1,i_2} \quad (1)$$

The total bandwidth resource capacities in satellite network $G(V, E)$ are $\sum_{e \in E} B_e$. Therefore, the normalized bandwidth cost for user request u can be described as:

$$\varphi_u^{bw} = \frac{u^{bw}}{\sum\limits_{e \in E} B_e} \quad (2)$$

According to the power consumption model [30], the energy consumption for an active edge server on satellite v is mainly produced by running CPU, which can be indicated as:

$$P_v^{on} = P_v^{idle} + \frac{\sum\limits_{u \in U} \sum\limits_{f_{u,i} \in F_u} x_{u,i}^v \cdot c_{u,i}^{cpu}}{C_v^{cpu}} \cdot (P_v^{max} - P_v^{idle}) \quad (3)$$

Considering that there are different energy consumptions for placing VNFs on an edge server in various running states, we divide the VNF placement into four solutions based on energy consumption costs as follows:

- *Case 1:* An edge server on satellite v is *available off* in the current time slot and will not provide computing services for any VNFs in the next time slot. In that case, when we place VNF $f_{u,i}$ on satellite v, the state

TABLE III
LIST OF SYMBOLS.

Satellite Network	
V	Set of satellites with the number of N.
v	The v-th satellite.
R	Set of resources offered by each satellite.
C_v^r	The r-th resource capacity for satellite v.
P_v^{idle}	Idle power of an edge server on satellite v.
P_v^{on}	Active power of an edge server on satellite v.
P_v^{max}	Maximum power of an edge server on satellite v.
t_v^{idle}	Maximum idle time of an edge server on satellite v.
t_v^{off}	Minimum off time of an edge server on satellite v.
E	Set of links between satellites.
e	The e-th link.
B_e	Bandwidth capacity for link e.
t_e	Transmission delay for link e.
$L_{v_1}^{v_2}$	Set of the d shortest paths between v_1 and v_2.
User Requests	
U	Set of M user requests.
u	The u-th user request.
F_u	Set of VNFs for user request u.
$f_{u,i}$	The i-th VNF for user request u.
s_u, d_u	Source and destination of user request u.
$c_{u,i}^r$	The r-th resource requirement for VNF $f_{u,i}$.
$t_{u,i}$	Execution time for VNF $f_{u,i}$.
H_u	Set of edges for user request u.
$h_u^{i_1,i_2}$	Edge between VNFs f_{u,i_1} and f_{u,i_2}.
$b_u^{i_1,i_2}$	Bandwidth resource requirement for $h_u^{i_1,i_2}$.
t_u^{delay}	Maximum acceptable delay time for user request u.
Binary Decision Variables	
$x_{u,i}^v$	$x_{u,i}^v = 1$ if $f_{u,i}$ is placed on satellite v.
$y_{u,l}^{i_1,i_2}$	$y_{u,l}^{i_1,i_2} = 1$ if path l is used by $h_u^{i_1,i_2}$.
Variables	
q_e^l	$q_e^l = 1$ if link e is used by path l, otherwise $q_e^l = 0$.
z_u	$z_u = 1$ if user request u is deployed, otherwise $z_u = 0$.
φ_u^{bw}	Bandwidth cost for user request u.
φ_u^{power}	Energy cost for user request u.
φ_u^{delay}	Delay cost for user request u.
φ_u	Payoff function for user request u.
Φ	Total payoff function.
α	Weight value.

of the edge server will be converted from *available off* to *on* during the setup procedure. Therefore, the power consumed by VNF $f_{u,i}$ in the current time slot can be denoted as $p_{u,i} = P_v^{max}$.

- *Case 2:* An edge server on satellite v is *available off* in the current time slot and will provide computing services for VNFs in the next time slot. In that case, when we place VNF $f_{u,i}$ on satellite v, according to the discussion in *Case 1*, the power consumed by VNF $f_{u,i}$ in the current time slot can be denoted as $p_{u,i} = 0$.
- *Case 3:* An edge server on satellite v is *idle* in the current time slot and will not provide computing services for any VNFs in the next time slot. Then if we place VNF $f_{u,i}$ on satellite v, due to the power consumption model [30], the power consumed by VNF $f_{u,i}$ in the current time slot can be denoted as $p_{u,i} = P_v^{idle} + \frac{x_{u,i}^v \cdot c_{v,i}^{cpu}}{C_v^{cpu}} \cdot (P_v^{max} - P_v^{idle})$.
- *Case 4:* An edge server on satellite v is *idle* or *on* in the current time slot and will provide computing services for VNFs in the next time slot. Then if we place VNF $f_{u,i}$ on satellite v, according to the discussion in *Case 3*, the power consumed by VNF $f_{u,i}$ in the current time slot can be denoted as $p_{u,i} = \frac{x_{u,i}^v \cdot c_{u,i}^{cpu}}{C_v^{cpu}} \cdot (P_v^{max} - P_v^{idle})$.

The total energy consumption in satellite network $G(V, E)$ is $\sum_{v \in V} P_v^{max}$, then the normalized energy cost for user request u can be expressed as:

$$\varphi_u^{power} = \frac{1}{\sum\limits_{v \in V} P_v^{max}} \sum_{f_{u,i} \in F_u} \sum_{v \in V} x_{u,i}^v \cdot p_{u,i} \tag{4}$$

For user request u, the service delay consists of computing delay and transmission delay, the computing delay can be denoted by:

$$t_u^{exec} = \sum_{f_{u,i} \in F_u} \sum_{v \in V} x_{u,i}^v \cdot t_{u,i} \tag{5}$$

The transmission delay for user request u can be indicated as:

$$t_u^{trans} = \sum_{h_u^{i_1,i_2} \in H_u} \sum_{v_1 \in V} \sum_{\substack{v_2 \in V \\ v_1 \neq v_2}} \sum_{l \in L_{v_1}^{v_2}} \sum_{e \in l} x_{u,i_1}^{v_1} \cdot x_{u,i_2}^{v_2} \cdot y_{u,l}^{i_1,i_2} \cdot q_e^l \cdot t_e \tag{6}$$

The maximum acceptable delay time for user request u is t_u^{delay}, which is the sum of the execution time for all VNFs and the acceptable path transmission time. In this paper, we assume that the acceptable path transmission time is equal to the average transmission time of all source-to-destination paths $L_{s_u,all}^{d_u}$ in satellite network $G(V, E)$. When $\left|L_{s_u,all}^{d_u}\right|$ indicates the number of source-to-destination paths in $L_{s_u,all}^{d_u}$, the maximum acceptable delay time for user request u can be denoted by:

$$t_u^{delay} = \sum_{f_{u,i} \in F_u} t_{u,i} + \frac{1}{\left|L_{s_u,all}^{d_u}\right|} \sum_{l \in L_{s_u,all}^{d_u}} \sum_{e \in l} t_e \tag{7}$$

Considering that the sum of the normalized service delay costs for all allocated user requests should be less than 1, the normalized service delay cost for user request u can be denoted by:

$$\varphi_u^{delay} = \frac{1}{\sum\limits_{u' \in U} z_{u'}} \cdot \frac{t_u^{exec} + t_u^{trans}}{t_u^{delay}} \tag{8}$$

For user request u, the deployment cost is the weighted sum of bandwidth cost φ_u^{bw}, energy consumption cost φ_u^{power}, and service delay cost φ_u^{delay}, where α_1, α_2, and α_3, $\alpha_1+\alpha_2+\alpha_3 = 1$, are the weighted factors accordingly and can be used to adjust the preferences of the three costs. Then the user payoff φ_u can be indicated by:

$$\varphi_u = (1 - \alpha_1 \cdot \varphi_u^{bw} - \alpha_2 \cdot \varphi_u^{power} - \alpha_3 \cdot \varphi_u^{delay}) \cdot z_u \tag{9}$$

The overall network payoff Φ is the sum of all user payoffs and can be represented by:

$$\Phi = \sum_{u \in U} \varphi_u \tag{10}$$

In order to address the problem of VNF placement to maximize the overall network payoff, the following physical constraints need to be considered.

When user request u is deployed to satellites, each VNF can be placed on one and only one satellite. We describe the VNF deployment constraint as follows:

$$\sum_{v \in V} x_{u,i}^{v} = z_u, \forall u \in U, \forall f_{u,i} \in F_u \tag{11}$$

For user request u, if two adjacent VNFs f_{u,i_1} and f_{u,i_2} are deployed on satellites v_1 and v_2, respectively, we need to guarantee that there exists one path between satellites v_1 and v_2 that can be used to route traffic flows from f_{u,i_1} to f_{u,i_2}. The path selection constraint can be expressed by:

$$\sum_{l \in L_{v_1}^{v_2}} y_{u,l}^{i_1,i_2} = x_{u,i_1}^{v_1} \cdot x_{u,i_2}^{v_2} \tag{12}$$

When we place VNFs for user requests to satellite nodes, the resource requirements for each satellite can not exceed its resource capacities. Then the satellite resource constraint can be described as:

$$\sum_{u \in U} \sum_{f_{u,i} \in F_u} x_{u,i}^{v} \cdot c_{u,i}^{r} \leq C_v^r, \forall v \in V, \forall r \in R \tag{13}$$

When we choose a path between two satellites to route traffic flows, the bandwidth requirements for each link should not be more than the bandwidth capacity. Then the bandwidth resource constraint for link e can be indicated by:

$$\sum_{u \in U} \sum_{h_u^{i_1,i_2} \in H_u} \sum_{v_1 \in V} \sum_{\substack{v_2 \in V \\ v_1 \neq v_2}} \sum_{l \in L_{v_1}^{v_2}} x_{u,i_1}^{v_1} \cdot x_{u,i_2}^{v_2} \cdot y_{u,l}^{i_1,i_2} \cdot q_e^l \cdot b_u^{i_1,i_2} \leq B_e \tag{14}$$

For user request u, the service delay time is also not more than the maximum acceptable delay time. The service delay time constraint can be described as:

$$t_u^{exec} + t_u^{trans} \leq t_u^{delay}, \forall u \in U \tag{15}$$

In addition, we need to ensure that the idle time $t_{v,gap}^{idle}$ for an edge server on satellite v can not exceed the maximum idle time t_v^{idle}. For an edge server on satellite v in the current *idle* state, we denote the earliest idle time by $t_{v,idle}^{old}$ and the current idle time by $t_{v,idle}^{new}$, respectively. The idle time constraint for an edge server on satellite v can be indicated by:

$$t_{v,gap}^{idle} = t_{v,idle}^{new} - t_{v,idle}^{old} \leq t_v^{idle}, \forall v \in V \tag{16}$$

We also guarantee that the off time $t_{v,gap}^{off}$ for an edge server on satellite v should be more than the minimum off time t_v^{off}. For an edge server on satellite v in the current *off* state, we use $t_{v,off}^{old}$ to indicate the earliest off time and $t_{v,off}^{new}$ to indicate the current off time, respectively. The off time constraint for an edge server on satellite v can be indicated by:

$$t_{v,gap}^{off} = t_{v,off}^{new} - t_{v,off}^{old} > t_v^{off}, \forall v \in V \tag{17}$$

Under the physical network resource and service requirement constraints in equations (11)-(17), the VNF placement problem in satellite edge computing can be formulated as an optimization problem with maximum network payoff, which can be indicated by:

$$\begin{aligned} max \quad & \Phi \\ s.t. \quad & (11) - (17) \end{aligned} \tag{18}$$

B. Problem Analysis

In this section, we reduce the capacitated plant location problem with single source constraints (CPLPSS) [33] to the above VNF placement problem and prove that the formulated VNF placement problem is NP-hard [22].

In CPLPSS, a set of potential locations is given for deploying plants with fixed capacities and costs, and goods from the plants need to be provided for a set of customers with fixed demands. There are transportation costs for supplying goods from the plants to the customers. In addition, the goods demanded by a customer are provided only by a single plant. The problem aims to find an optimal strategy of placing plants within the capacity and demand constraints to minimize the total operational and transportation costs.

To reduce a CPLPSS problem to the proposed VNF placement, we re-construct the problem description. active edge servers on satellites indicate plants and satellite resources represent plant capacities. User requests can be viewed as customers and resource demands for their computation tasks are described as customer required goods. We assume that all demanded resources from a user request are offered only by a satellite node. The operational cost of a plant is denoted by energy consumption and the transportation cost can be indicated by bandwidth and source-to-destination delay costs. For the proposed VNF placement problem, maximum overall network payoff is equal to minimizing the sum cost of energy consumption, bandwidth, and service delay costs within the constant number of user requests. Thus, a CPLPSS problem can be reduced to the proposed optimization problem. As a CPLPSS problem is NP-hard, the VNF placement problem is also NP-hard.

V. VNF Placement Game and Proposed Algorithm

In this section, we formulate the VNF placement problem as a potential game and analyze its property by a game-theoretical approach. Then a decentralized resource allocation algorithm based on a potential game is proposed for addressing the VNF placement problem.

A. VNF Placement Game

Game theory is a mathematical tool for analyzing interactive decision-making processes. A non-cooperative game can be denoted by $\Gamma = \{U, A, \varphi\}$. The term U indicates the set of players. The strategy of player u is denoted by a_u, $a_u \in A_u$, where A_u represents the strategy set of player u. $A = \prod_{u \in U} A_u$, denotes the strategy combination of all players and $a = \{a_u | u \in U\}$ indicates the strategies of all players. We denote the strategies of all players except player u as $a_{-u} = \{a_{u'} | u' \in U, u' \neq u\}$ and the strategy combination of all players except player u as $A_{-u} = \prod_{u' \in U, u' \neq u} A_{u'}$. The term $\varphi_u(a_u)$ indicates the payoff of player u with the strategy a_u and the term $\varphi(a) = \{\varphi_u(a_u) | u \in U\}$ is the payoff set of all players.

For a non-cooperative game, the players can make their strategies in a self-interested way for maximizing their payoffs until a Nash equilibrium is generated, where each player can not unilaterally deviate its strategy for improving the payoff.

Algorithm 1 Resource Allocation Based on a Potential Game.

Input: User requests U;
Output: $a^* = \{a_u^* | u \in U\}$;

Initialize: $k = 0$, $a_u(k) = \varnothing$ for each user request u;
while $k < K_{max}$ **do**
  for each $u \in U$ in parallel **do**
    for each $l \in L_{s_u}^{d_u}$ **do**
      Search an optimal strategy $a'_u(k,l)$ for path l by the Viterbi algorithm and compute the user payoff $\varphi(a'_u(k,l), a_{-u}(k))$;
    end for
    Find the strategy $a'_u(k)$ with maximum user payoff according to $a'_u(k) = \arg \max_{a'_u(k,l), l \in L_{s_u}^{d_u}} \varphi(a'_u(k,l), a_{-u}(k))$;
    if the user payoff of $a'_u(k)$ for u is improved **then**
      Share strategy $a'_u(k)$ with other user requests;
    end if
  end for
  Run a competitive mechanism for all user requests and update strategy $a'(k)$ with strategy $a'_u(k)$;
  if $|\Phi(a'(k)) - \Phi(a(k))| < \epsilon$ **then**
    $a^* = a'(k)$ and break;
  else
    $k \leftarrow k + 1$;
  end if
end while
return a^*;

Definition 1 *(**Nash equilibrium**) A strategy profile $a^* = \{a_u^* | u \in U\}$ of all players is a Nash equilibrium if no player has an incentive for unilaterally deviating the strategy, such that,*

$$\varphi_u(a_u^*, a_{-u}^*) \geq \varphi_u(a_u, a_{-u}^*), \forall u \in U, \forall a_u \in A_u \tag{19}$$

Before finding a Nash equilibrium of the game, we should ensure whether the game admits at least a Nash equilibrium. As a special instance of non-cooperative games, potential game possesses a pure strategy Nash equilibrium [20], where a global potential function is used to map the payoffs of all players. The game can be considered as an exact potential game if an exact potential function is admitted.

Definition 2 *(**Exact Potential Game**) A game is an exact potential game if, for an exact potential function $\Phi(a), u \in U, a_u, a'_u \in A_u$, and $a_{-u} \in A_{-u}$, there is,*

$$\Phi(a'_u, a_{-u}) - \Phi(a_u, a_{-u}) = \varphi_u(a'_u, a_{-u}) - \varphi_u(a_u, a_{-u}) \tag{20}$$

The key for the existence of a Nash equilibrium is to prove the VNF placement game is a potential game. For the VNF placement in satellite edge computing, user requests have potential conflicts in maximizing their payoffs and the strategy of a user request has an effect on that of other user requests. We formulate the VNF placement problem as a non-cooperative game, where M user requests are M players, the VNF placement strategy for user request u represents the strategy a_u and the payoff for user request u indicates the payoff of player u. The term A_u indicates the strategy set of user request u and A is the strategy combination of all user requests. The exact potential function is denoted by $\Phi(a)$, which is the sum of all user payoffs. Then we prove that the VNF placement game is a potential game.

Proposition 1 *The VNF placement game is an exact potential game, where the payoff of player u is indicated by $\varphi_u(a_u)$ and the exact potential function is the network payoff $\Phi(a)$.*

Proof *For $a'_u, a_u \in A_u$, $a_{-u} \in A_{-u}$, according to equation* (9)*, there is,*

$$\Delta\varphi_u = \varphi_u(a'_u, a_{-u}) - \varphi_u(a_u, a_{-u}) \tag{21}$$

Similarly, according to equation (10)*, we can obtain the potential function difference as:*

$$\begin{aligned}\Delta\Phi &= \Phi(a'_u, a_{-u}) - \Phi(a_u, a_{-u}) \\ &= \left[\varphi_u(a'_u, a_{-u}) + \sum_{\substack{u' \in U \\ u' \neq u}} \varphi_{u'}(a_{u'})\right] - \left[\varphi_u(a_u, a_{-u}) + \sum_{\substack{u' \in U \\ u' \neq u}} \varphi_{u'}(a_{u'})\right] \\ &= \varphi_u(a'_u, a_{-u}) - \varphi_u(a_u, a_{-u})\end{aligned} \tag{22}$$

There is $\Delta\Phi \equiv \Delta\varphi_m$ [21], [34]. Thus, we prove that the VNF placement game is an exact potential game and $\Phi(a)$ is an exact potential function.

Considering that the VNF placement game is a potential game and has the finite improvement property [35], the VNF placement problem can be addressed by finding a Nash equilibrium in a finite gradual iteration. A decentralized resource allocation algorithm based on a potential game, which is discussed in the following subsection, is implemented to tackle the VNF placement problem.

B. Decentralized Resource Allocation Algorithm

As the VNF placement problem is NP-hard [22], we implement a decentralized resource allocation algorithm by a potential game (PGRA) to find an approximate strategy. The proposed PGRA algorithm can perform on the satellite network to improve the real-time decision-making capacity, where multiple satellites share the network resource states and the VNF placement strategies by interacting with each other, but satellites does not need to exchange the information with IoT users during the running time of the proposed PGRA algorithm. The procedure of the proposed PGRA algorithm is shown in Algorithm 1. The maximum number of iterations is K_{max}. At the beginning, the initial iteration time is $k = 0$, for each user request u, we initialize the strategy as $a_u(k) = \varnothing$ and the user payoff as $\varphi_u(a_u(k), a_{-u}(k))$ accordingly. In iteration time k, user request u needs to find an optimal strategy $a'_u(k)$ with maximum user payoff $\varphi_u(a'_u(k), a_{-u}(k))$ while the strategies of other user requests remain unchanged. For obtaining the optimal strategy of user request u, we search the d shortest paths between source s_u and destination d_u successively and the local optimal strategy $a'_u(k,l)$ for each path l is calculated by the Viterbi algorithm [36], which will

Algorithm 2 Viterbi Algorithm.

Input: u, l;
Output: $a'_u(k,l)$;

Initialize: $a'_u(k,l) = \varnothing, A_{layer} = \varnothing$;
Obtain topological sort sequence Γ_u for the VNFs;
for each $f_{u,i} \in \Gamma_u$ **do**
  $A'_{layer} = \varnothing$;
  if $f_{u,i} = s_u$ **then**
    $A_{layer} \leftarrow$ configure information about s_u;
    continue;
  end if
  for each $a^{pre}_{u,i}(k) \in A_{layer}$ **do**
    Update network resource conditions under $a^{pre}_{u,i}(k)$;
    Obtain the set $\Omega_{u,i}$ of available satellites for path l;
    for each satellite $v \in \Omega_{u,i}$ **do**
      Deploy $f_{u,i}$ to satellite v by strategy $a^{curr}_{u,i}(k)$;
      if the constraints in (11)-(17) are satisfied **then**
        $A'_{layer} \leftarrow \left[a^{pre}_{u,i}(k), a^{curr}_{u,i}(k)\right]$;
      end if
    end for
  end for
  List A'_{layer} by user payoffs in descending order.
  $A_{layer} \leftarrow A'_{layer}[:B]$;
end for
Obtain strategy $a'_u(k,l)$ with maximum user payoff;
return $a'_u(k,l)$;

be discussed later. Thus we can acquire an optimal strategy of user request u in the current iteration by:

$$a'_u(k) = \arg \max_{a'_u(k,l), l \in L^{d_u}_{s_u}} \varphi(a'_u(k,l), a_{-u}(k)) \qquad (23)$$

If the network payoff of $a'_u(k)$ is improved compared to $a_u(k)$, the strategy $a'_u(k)$ will be shared by a message synchronization mechanism to other user requests for competing available resources of the satellite network. In each iteration, only a user request, which makes the overall network payoff maximum, can win the opportunity for updating its decision-making strategy and other user requests need to keep their old decision-making strategies. Note that the strategy decision-making processes for all user requests are performed simultaneously in parallel. The iteration process will terminate when no user request has an incentive to deviate its strategy unilaterally or the number of iterations exceeds the maximum iteration threshold. The final strategy profile $a^* = \{a^*_u | u \in U\}$ for all user requests is a Nash equilibrium of the VNF placement game and represents an approximate VNF placement strategy.

In this paper, we search the d shortest source-to-destination paths for user request u, $u \in U$, and use the Viterbi algorithm to place the VNFs for each path. The Viterbi algorithm can be viewed as a multi-stage graph for the states and their relationships and its aim is to find the most likely sequence of the states. The number of stages is equal to the length of an observed event sequence. Each node in a stage represents a possible state with a fixed cost for the current observed event and each stage in a graph consists of all possible states for the current observed event. The weighted edge between two

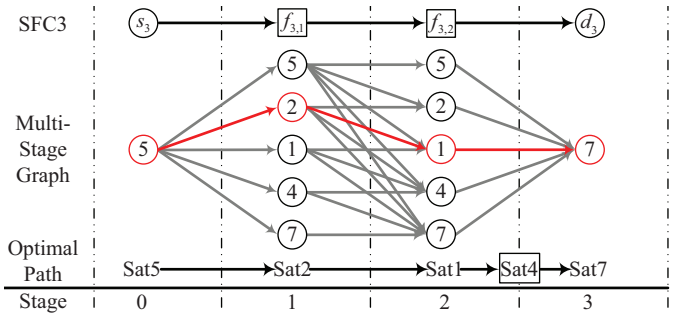


Fig. 4. Example of placing VNFs for a user request by the Viterbi algorithm.

states from adjacent stages represents a transmission cost. We can compute the cumulative cost of each state by the Viterbi algorithm in an increasing stage order, where the cumulative cost is composed of fixed costs and transmission costs. In the final stage, there are multiple possible states and each possible state corresponds to a path with a cumulative cost. We select a state path with minimum cumulative cost as the most likely path and obtain the best sequence by tracing the path. As the number of observed events and states increases the search space becomes large. Therefore, we can cut the search tree width to reduce the computational complexity.

For finding an approximate strategy of the VNF placement by the Viterbi algorithm, we construct the VNF placement states and their relationships as a multi-stage graph. For a user request, the SFC is considered as an observed event sequence, each VNF indicates an observed event and the number of the VNFs represents the number of stages. The strategy of deploying a VNF indicates a possible state in each stage and there has the VNF deployment cost accordingly. The path between two adjacent VNFs indicates the edge between two states from two adjacent stages and the path cost is equal to the edge cost. Thus, we can perform the Viterbi algorithm to obtain an approximate strategy of VNF placement for a user request. Note that when the VNFs for a user request are deployed to satellites, the repeated path between two satellites will be not allowed in order to decrease the deployment cost of delay and bandwidth. That is, if VNF $f_{u,i}$ for user request u is deployed to satellite v, then the available satellites for the successor of VNF $f_{u,i}$ consist of satellites in the remaining path from satellite v to the destination satellite.

Fig. 4 illustrates an example of placing VNFs for a user request by the Viterbi algorithm, where the user request and the VNF placement strategy are shown in Table II. At the beginning, source s_3 is located on satellite $Sat5$ and we can obtain an initial user payoff in stage 0. Then we begin to deploy VNF $f_{3,2}$ to available satellites and calculate the cumulative cost for each possible state in stage 1, where the available satellites are $\{Sat5, Sat2, Sat1, Sat4, Sat7\}$ as the discussion earlier. After deploying VNF $f_{3,2}$, we reserve B possible states with maximum user payoffs into stage 2 in order to reduce the computation complexity. Similar the procedure in stage 1, we deploy VNF $f_{3,3}$ for each reserved state to available satellites, where when VNF $f_{3,2}$ is deployed to satellite $Sat2$, the available satellites for the possible state

TABLE IV
SIMULATION PARAMETERS.

Satellite Networks [37]–[39]							
Name	Total Number of Satellites			Inter-Satellite Link Distance		Inter-Satellite Link Bandwidth	
Value	6,9,12,15			400 km,600 km		100 Mbps	
Edge Servers on Satellites [40], [41]							
Name	vCPUs	Memory	Idle Power	Maximum Active Power	Setup Power	Maximum Idle Time	Minimum Off Time
Value	112	192 GB	49.9 W	415 W	415 W	3 slots	1 slot
User Requests [36], [41]							
Name	VNFs	vCPUs	Memory	Execution Time for VNFs	Bandwidth	Running Time	
Lower	5	4	4 GB	10 ms	10 Mbps	1 slot	
Upper	10	8	16 GB	30 ms	30 Mbps	4 slots	

will be $\{Sat2, Sat1, Sat4, Sat7\}$. In stage 3, destination d_3 is located satellite $Sat7$. Then if user request u is deployed to satellites, we can obtain multiple paths with user payoffs, where the path with maximum user payoff is considered as the most likely path and we can trace the path to find an approximate VNF placement strategy, as shown with a red line in Fig. 4.

The Viterbi algorithm for the VNF placement is shown in Algorithm 2. The input parameters are user request u and path l. The output parameter is an approximate strategy $a'_u(k,l)$ for path l. Initially, we denote a state set of the first stage by $A_{layer} = \varnothing$ and set $a'_u(k,l) = \varnothing$. Then we can obtain the ordered VNF sequence Γ_u by a topological sorting method. For each stage of VNF $f_{u,i} \in \Gamma_u$, we search all possible VNF placement states and calculate their cumulative costs, respectively. If $f_{u,i} = s_u$, we can directly update the configure information concerning s_u to A_{layer}. If $f_{u,i} \neq s_u$, for $a^{pre}_{u,i}(k)$, $a^{pre}_{u,i}(k) \in A_{layer}$, we first update network resource conditions by $a^{pre}_{u,i}(k)$ and obtain a set $\Omega_{u,i}$ of current available satellites for path l. Within the network resource and service requirement constraints, we deploy VNF $f_{u,i}$ to satellite v, $v \in \Omega_{u,i}$, by strategy $a^{curr}_{u,i}(k)$. When the constraints in equations (11)-(17) are satisfied, $a^{curr}_{u,i}(k)$ will be put into a new strategy set A'_{layer}, which is initialized as $A'_{layer} = \varnothing$ at the beginning of each stage. When each stage is over, we will sort A'_{layer} by user payoffs in descending order and use the first B paths from A'_{layer} to update A_{layer}. If all VNFs are deployed we can obtain an approximate strategy $a'_u(k,l)$ with maximum user payoff.

For Algorithm 2, we assume that the search tree width is B, the number of satellite nodes for an available path is less than N, and the maximum number of VNFs for a user request is F, then the computation complexity can be described as $O(FBN)$. For Algorithm 1, the computation complexity can be indicated as $O(K_{max}MdFBN)$, where K_{max} is the maximum number of iterations, M is the number of players for the current time slot, and d represents the size of a candidate path set.

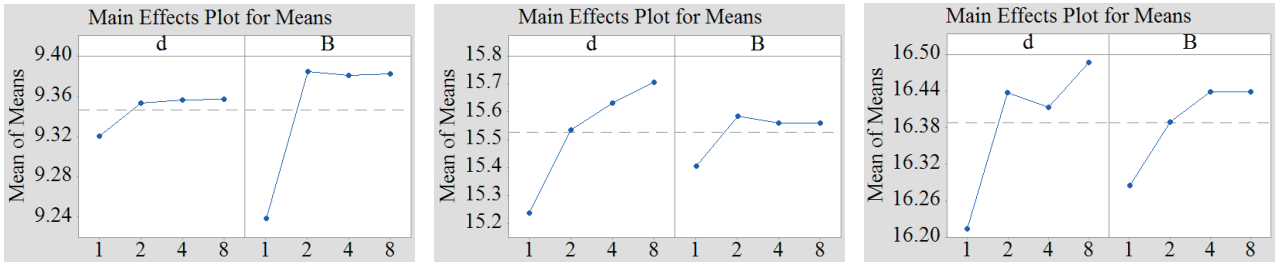
VI. PERFORMANCE EVALUATION

In this section, we make the experiments to evaluate the performance of the proposed PGRA algorithm for addressing the VNF placement problem in satellite edge computing. We setup the system parameters for performance evaluation. In order to investigate the effects of system parameters on the performance of the proposed PGRA algorithm, we design the experiments by the Taguchi method with two factors, which are the shortest paths d between the source and the destination for a user request and the width B of the Viterbi search tree. Furthermore, we compare the proposed PGRA algorithm with the existing centralized baseline algorithms of Viterbi [22], Greedy [42], and Gurobi Optimizer [43] in terms of network payoff and percentage of allocated users. Finally, we discuss the performance of the proposed PGRA algorithm in on-line strategy.

A. Simulation Setup

In the simulation experiments, we set the number of LEO satellite nodes by 6, 9, 12, and 15, respectively, where the number of orbital planes is 2, 3, 4, and 5, respectively, and each orbital plane has 3 satellites. There is an edge server on each satellite. We configure the resource capacities for each edge server as 112 vCPUs and 192 GB Memory, the idle power as 49.9 W, and the maximum active power as 415 W [40]. We assume that the maximum idle time interval is 3 time slots and the minimum off time interval is 1 time slot [41]. In addition, the initial inter-satellite link distance for different orbital planes is considered as 600 km and 400 km, respectively. The bandwidth capacity for each inter-satellite link is set as 100 Mbps. In dynamic environment, we use Systems Tool Kit (STK) [37] to obtain the satellite orbital elements, use python package *SGP4* [38] to simulate the mobility of satellites over different time slots, and then build the satellite networks with dynamic characteristics by python package *Networkx* [39].

In order not to lose generality, we randomly generate VNFs and resource requirements for each user request [36], [41]. Specifically, the number of VNFs is from 5 to 10. Each VNF, except source and destination, has a predecessor and a successor, and requires $[4, 8]$ vCPUs and $[4, 16]$ GB Memory, respectively. The execution time for each VNF is $[10, 30]$ ms. The bandwidth demand for each edge is $[10, 30]$ Mbps. The source and the destination are randomly generated from the set of satellite nodes and can be known in advance. In addition, we define the weighted values in equation (9) as $\alpha_1 = \alpha_2 = \alpha_3 = \frac{1}{3}$. Table IV summarizes the main simulation parameters in the evaluation. We use a commodity server to be the simulation platform, where the configuration information is i7-4790K CPU, 16 GB Memory, and Windows 10. The programming language is PYTHON.



(a) Main effects for $M = 10$ (b) Main effects for $M = 20$ (c) Main effects for $M = 30$

Fig. 5. Main effects of two factors for different number $M = 10, 20$, and 30 of user requests.

TABLE V
PARAMETERS FOR TAGUCHI METHOD.

Factor	Level			
	1	2	3	4
d	1	2	4	8
B	1	2	4	8

TABLE VI
ORTHOGONAL TABLE $L_{16}(4^2)$ AND NETWORK PAYOFF RESULTS.

Number	Factor		Network Payoff		
	d	B	M=10	M=20	M=30
0	1	1	9.1585	15.1844	16.1857
1	1	2	9.3734	15.1888	16.2896
2	1	4	9.3748	15.2886	16.1899
3	1	8	9.3751	15.2891	16.1899
4	2	1	9.2622	15.3828	16.3832
5	2	2	9.3822	15.5843	16.3890
6	2	4	9.3832	15.5844	16.4879
7	2	8	9.3846	15.5844	16.4879
8	4	1	9.2680	15.4805	16.2845
9	4	2	9.3916	15.6820	16.3895
10	4	4	9.3825	15.6821	16.4884
11	4	8	9.3838	15.6814	16.4884
12	8	1	9.2680	15.5785	16.2846
13	8	2	9.3919	15.8785	16.4872
14	8	4	9.3828	15.6819	16.5862
15	8	8	9.3840	15.6812	16.5862

B. System Parameters Evaluation

For the proposed PGRA algorithm, the performance results of addressing the VNF placement problem can be influenced by two important parameters, which are the shortest paths d between the source and the destination for a user request and the width B of the Viterbi search tree, respectively. Consequently, in a satellite network with 6 satellite nodes, we use the Taguchi method of design-of-experiment (DOE) [44] to investigate the simulation results under different values of d and B [45]. The two factors are denoted by d and B, each factor includes 4 levels, i.e., 1, 2, 4, and 8, as shown in Table V. The orthogonal table $L_{16}(4^2)$ consists of 16 instances and we run each instance, for M = 10, 20, and 30, 10 times to obtain the average network payoffs, respectively. Table VI describes the orthogonal table and network payoff results. The main effects of the two factors for different user requests are illustrated in Fig. 5, where mean of means indicates the average network payoff results for all cases concerning the specific level of one factor. We can find from Fig. 5 that the network payoff results for M = 10, 20, and 30 are better as parameters d and B increase. Large d can improve the exploration ability of the proposed PGRA algorithm by searching for a larger strategy space. For the Viterbi algorithm, large B can keep more possible states of the current stage to the next search stage and also promote the network performance. However, large d and B can lead to a high computational complexity of the proposed PGRA algorithm. When d and B are small, the performance of the proposed PGRA algorithm will degrade. Therefore, their values should be considered in a tradeoff way, where we can find that the ideal values of d and B are 8 and 4 according to the Taguchi method.

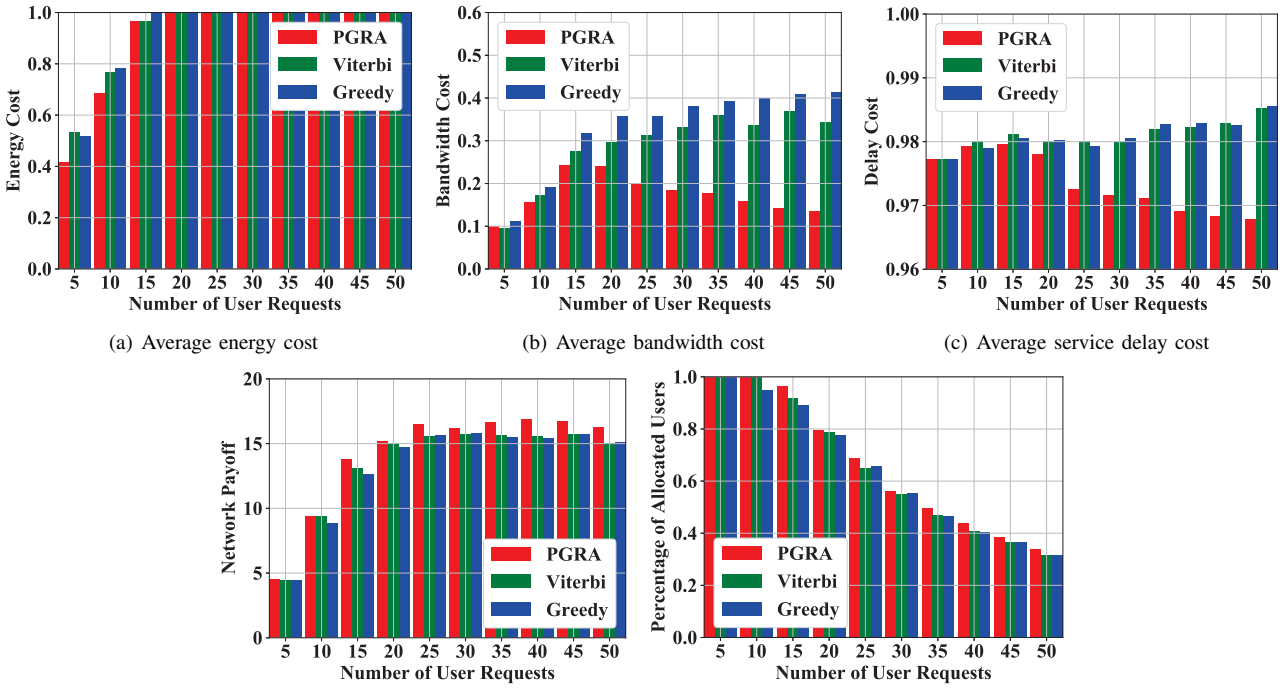
TABLE VII
SIMULATION RESULTS FOR DIFFERENT WEIGHTED FACTORS

Name		$[\frac{3}{8}, \frac{1}{8}, \frac{1}{2}]$	$[\frac{1}{8}, \frac{1}{2}, \frac{3}{8}]$	$[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$
Energy Cost	M=10	0.6666	0.6666	0.6666
	M=20	1	1	1
	M=30	1	1	1
Bandwidth Cost	M=10	0.108	0.107	0.108
	M=20	0.1618	0.1645	0.1618
	M=30	0.1429	0.1471	0.1429
Delay Cost	M=10	0.9964	0.9965	0.9964
	M=20	0.9962	0.9962	0.9962
	M=30	0.9959	0.9959	0.9959
Network Payoff	M=10	9.3779	9.2795	9.4096
	M=20	15.6161	15.5058	15.5806
	M=30	16.4234	16.3081	16.387
Percentage of Allocated Users	M=10	1	1	1
	M=20	0.8149	0.8199	0.8149
	M=30	0.5699	0.5733	0.5699

To evaluate the impact of weighted factors in equation (9) on the performance of the proposed PGRA algorithm, we conduct the experiments with d = 8 and B = 4 in three scenarios of different weighted factors, where the values of weighted factors, α_1, α_2, and α_3, can be indicated as $[\frac{3}{8}, \frac{1}{8}, \frac{1}{2}]$, $[\frac{1}{8}, \frac{1}{2}, \frac{3}{8}]$, and $[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$, respectively. The results of the proposed PGRA algorithm for different weighted factors are shown in Table VII. We can find from Table VII that different preferences of the deployment costs have an impact on the network payoffs as well as the VNF placement strategies, where the preferences of the deployment costs, i.e., energy consumption, network bandwidth, and service end-to-end delay, can be adjusted by changing the values of the weighted factors.

C. Performance Comparison with Baseline Algorithms

To further discuss the effectiveness of the proposed PGRA algorithm in terms of energy consumption, bandwidth, and



(a) Average energy cost (b) Average bandwidth cost (c) Average service delay cost

(d) Average network payoff (e) Average percentage of allocated users

Fig. 6. Performance comparison with PGRA, Viterbi, and Greedy in a satellite network with 6 satellite nodes.

service delay, we compare the proposed PGRA algorithm with two existing baseline algorithms of Viterbi [22] and Greedy [42].

Viterbi: The Viterbi algorithm in [22] is introduced to address the VNF placement problem, where the Viterbi algorithm is also considered as part of the proposed PGRA algorithm, as shown in Algorithm 2. During the VNF placement, we seek the approximate VNF placement strategies by the Viterbi algorithm for user requests one by one.

Greedy: We use the Greedy algorithm [42] to tackle the VNF placement problem. For each available path, the VNFs from a user request in topological order can be deployed to satellites one by one. There are multiple available satellites for each VNF, where we consider the VNF placement strategy with minimum deployment cost of energy consumption and network bandwidth as the current optimal strategy. Note that only one VNF placement strategy for each VNF can remain into the next VNF placement stage.

Based on the parameter analysis of the Taguchi method, the values of d and B are set as 8 and 4, respectively. A satellite network with 6 satellite nodes is used to run 10 group experiments and the corresponding number of user requests is denoted by $M = \{5, 10, 15, 20, \cdots, 50\}$. Each experiment is run 10 times and the average results are obtained.

The experiment results for the performance comparison with three optimization algorithms are shown in Fig. 6. Fig. 6(a) illustrates the average energy costs for different number of user requests. We can find from Fig. 6(a) that the energy costs obtained by the proposed PGRA algorithm are better than that of Viterbi and Greedy when there is a small number of user requests, e.g., $M = 5$, 10, and 15. For $M = 10$, the average energy costs for PGRA, Viterbi, and Greedy are 0.6833, 0.7666, and 0.7833, respectively. The average energy cost obtained by the proposed PGRA algorithm reduces by 10.87% for Viterbi and 12.77% for Greedy. However, as the number of user requests increases, the energy costs are increasing until they reach the maximum values. Then new user requests can not be deployed to satellite nodes due to the limitation of network resource capacities.

The average bandwidth costs for deploying different user requests to satellite nodes are described in Fig. 6(b). We can observe from Fig. 6(b) that the proposed PGRA algorithm performs better than the two baseline algorithms of Viterbi and Greedy. For the small number of user requests, such as $M = 5$, 10, and 15, the performance of the proposed PGRA algorithm is slightly better than that of Viterbi and Greedy. For an example of $M = 10$, the average bandwidth costs are 0.1560, 0.1711, and 0.1915 for PGRA, Viterbi, and Greedy, respectively. The performance improvement of the proposed PGRA algorithm is 8.82% for Viterbi and 18.53% for Greedy. As the number of user requests increases, we can observe that the performance differences between the proposed PGRA algorithm and the two baseline algorithms are also increasing. In the case of $M = 35$, the average bandwidth costs for PGRA, Viterbi, and Greedy are 0.1756, 0.3581, and 0.3923, respectively. We can observe that the performance of the proposed PGRA algorithm improves by 50.96% for Viterbi and 55.23% for Greedy. That is due to the fact that the players in a potential game want to make decisions by competing with each other for optimizing their objectives in a self-interested way. In a resource constrained scenario, the players, which have better strategies of VNF placement, can win the opportunities of updating their strategy information. Therefore, the higher is the user payoff for a

TABLE VIII
PERFORMANCE COMPARISON WITH PGRA, VITERBI, AND GREEDY IN DIFFERENT SATELLITE NETWORKS.

Name		Energy Cost			Bandwidth Cost			Delay Cost			Network Payoff			Percentage of Allocated Users		
		N=9	N=12	N=15	N=9	N=12	N=15	N=9	N=12	N=15	N=9	N=12	N=15	N=9	N=12	N=15
PGRA	M=10	0.5	0.3583	0.2933	0.1185	0.124	0.1065	0.962	0.9473	0.9301	9.4731	9.5234	9.5566	1	1	1
	M=20	0.8555	0.675	0.5333	0.2338	0.1992	0.1829	0.9602	0.9465	0.9269	19.2167	19.393	19.4522	0.995	1	1
	M=30	1	0.925	0.76	0.2646	0.3184	0.2612	0.9579	0.9441	0.9232	24.2591	28.7708	29.3518	0.8333	0.9833	1
	M=40	1	1	0.9933	0.2558	0.3407	0.4006	0.9565	0.9415	0.9282	24.7625	33.3392	38.9259	0.6375	0.8525	0.9925
Viterbi	M=10	0.5888	0.4666	0.4	0.1216	0.179	0.0965	0.961	0.9426	0.9274	9.4428	9.4942	9.5253	1	1	1
	M=20	0.9222	0.7916	0.6466	0.2767	0.2276	0.1956	0.961	0.9457	0.9253	19.28	19.3449	19.4107	1	1	1
	M=30	1	0.9833	0.8333	0.384	0.3709	0.2823	0.9659	0.9463	0.9227	23.6466	28.8331	29.3205	0.8133	0.9866	1
	M=40	1	1	1	0.4098	0.4463	0.4413	0.9688	0.9472	0.9298	23.207	32.4021	38.2096	0.6	0.83	0.975
Greedy	M=10	0.5444	0.4583	0.4	0.1467	0.124	0.1178	0.962	0.9434	0.9288	9.1489	9.3913	9.4177	0.97	0.99	0.99
	M=20	0.9222	0.7333	0.6333	0.311	0.2652	0.2167	0.9611	0.947	0.9255	18.9685	19.2514	19.2081	0.985	0.995	0.99
	M=30	1	0.95	0.8133	0.4283	0.4291	0.3273	0.9649	0.9469	0.9232	23.9022	28.4246	28.612	0.8233	0.9733	0.9766
	M=40	1	1	1	0.4667	0.5101	0.5022	0.9685	0.949	0.9309	23.5882	32.3802	37.3889	0.61	0.83	0.955

player, i.e., the lower is the deployment cost, the greater is the winning chance. When more user requests ask for computing services, these user requests with maximum user payoffs can be deployed to satellites for improving the overall network payoff, where all edge servers are *on* to provide computing services for user requests. In that case, note that the energy costs for user requests remain unchanged and the user payoffs are mainly affected by network bandwidth and service end-to-end delay. Therefore, as the number of user requests increases, we can find that the network bandwidth costs obtained by the proposed PGRA algorithm are better than that of Greedy and Viterbi. On the other hand, when there are more user requests to deploy to satellites, the number of VNF placement strategies will increase and then the better VNF placement strategies with minimum user payoffs may be obtained. That is the reason why the bandwidth costs are gradually decreasing as the number of user requests increases, as shown in Fig. 6(b). However, the average performance improvement of the proposed PGRA algorithm is 40.05% for Viterbi and 47.93% for Greedy.

In Fig. 6(c), average service delay costs for different user requests are indicated. Similar to the average bandwidth costs in Fig. 6(b), we can observe that the average service delay costs obtained by the proposed PGRA algorithm are better than that of the two baseline algorithms, where the service delay costs are also gradually decreasing when the number of user requests increases in a resource constrained scenario. On average, the service delay costs for PGRA, Viterbi, and Greedy are 0.9734, 0.9811, and 0.9810, respectively. The performance of the proposed PGRA algorithm improves by 0.78% over both Viterbi and Greedy.

Fig. 6(d) describes the average network payoffs for different number of user requests. It can be found from Fig. 6(d) that the proposed PGRA algorithm outperforms the two baseline algorithms of Viterbi and Greedy in all the experiments. For an example of M = 15, the network payoffs for PGRA, Viterbi, and Greedy are 13.7703, 13.0588, and 12.6343, respectively, and the result obtained by the proposed PGRA algorithm is over 5.44% for Viterbi and 8.99% for Greedy. Overall, the average performance improvement of the proposed PGRA algorithm is 5.16% for Viterbi and 6.15% for Greedy, respectively. In Fig. 6(e), we illustrate the average percentages of allocated user requests. We can find from Fig. 6(e) that all user requests can be deployed to satellite nodes when the number of user requests is small, e.g., M = 10. As M increases, the percentage of allocated user requests begins to decrease due to the resource limitation of a satellite network. In these cases, the proposed PGRA algorithm also outperforms Viterbi and Greedy in terms of the percentage of allocated user requests. On average, the performance of the proposed PGRA algorithm is better 3.18% than Viterbi and 4.60% than Greedy.

In order to further evaluate the performance of the proposed PGRA algorithm in different scale satellite networks, three satellite networks, which consist of N = 9, 12, and 15 satellite nodes, are used for conducting the experiments with $M = \{10, 20, 30, 40\}$, respectively. Each experiment is run 10 times and Table VIII shows the average results in terms of energy consumption, network bandwidth, end-to-end delay, network payoff, and percentage of allocated user requests. In all cases, the proposed PGRA algorithm performs better than Greedy and Viterbi for energy costs. In most of the cases, the proposed PGRA algorithm performs better than Greedy and Viterbi in terms of network bandwidth and service end-to-end delay. However, on average, the proposed PGRA algorithm outperforms Greedy and Viterbi for the overall network payoffs and the percentages of allocated user requests.

For an example of N = 12, on average, the energy cost obtained by the proposed PGRA algorithm can reduce by 8.73% for Viterbi and 5.83% for Greedy, while the proposed PGRA algorithm can save the network bandwidth cost by 19.73% for Viterbi and 26.05% for Greedy, respectively. The performance improvement of the proposed PGRA algorithm for the service end-to-end delay is 0.06% for Viterbi and 0.18% for Greedy. Besides, the proposed PGRA algorithm improves the network payoff by 1.05% for Viterbi and 1.76% for Greedy, and increases the percentage of allocated user requests by 0.50% for Viteri and 1.25% for Greedy, respectively. On the other hand, we can find that more network resources are available to provide computing services for user requests as the number of satellites increases, therefore, the network payoffs and the percentages of allocated user requests increase accordingly. In the case of M = 40, the network payoffs obtained by the proposed PGRA algorithm for N = 9, 12, and 15 are 24.7625, 33.3392, and 38.9259, and the percentages of allocated user requests are 63.75%, 85.25%, and 99.25%, respectively. From Fig. 6 and Table VIII, we can demonstrate the effectiveness of the proposed PGRA algorithm compared with two baselines of Viterbi and Greedy, meanwhile, it is

TABLE IX
PERFORMANCE COMPARISON WITH PGRA AND GUROBI IN DIFFERENT SATELLITE NETWORKS.

Name		Energy Cost		Bandwidth Cost		Delay Cost		Network Payoff		Percentage of Allocated Users	
		PGRA	Gurobi	PGRA	Gurobi	PGRA	Gurobi	PGRA	Gurobi	PGRA	Gurobi
N=6	M=6	0.5167	0.45	0.0964	0.0945	0.9785	0.9789	5.4695	5.4922	1	1
	M=8	0.6167	0.5333	0.1151	0.1274	0.9773	0.9787	7.4303	7.4535	1	1
	M=10	0.7167	0.6833	0.1418	0.1336	0.9791	0.9794	9.3875	9.4012	1	1
	M=12	0.8333	0.8	0.1949	0.1745	0.9804	0.9804	11.2305	11.3484	0.9917	1
	M=14	0.9167	0.8833	0.1918	0.1811	0.9784	0.9782	13.0044	13.3191	0.9786	1
	M=16	1	1	0.2332	0.232	0.9801	0.9811	14.5622	15.1623	0.9563	0.9938
N=9	M=6	0.3333	0.3222	0.0773	0.07	0.9637	0.9629	5.5419	5.5483	1	1
	M=8	0.4111	0.3667	0.1038	0.1048	0.9631	0.9635	7.5073	7.5217	1	1
	M=10	0.4778	0.4556	0.1272	0.111	0.9635	0.9618	9.4772	9.4905	1	1
	M=12	0.5556	0.5333	0.1538	0.1348	0.9631	0.9621	11.4425	11.4566	1	1
	M=14	0.6333	0.5889	0.1609	0.1415	0.9604	0.9598	13.4151	13.4366	1	1
	M=16	0.7111	0.7	0.1928	0.1534	0.9618	0.9607	15.3781	15.3953	1	1

shown that the proposed PGRA algorithm outperforms Viterbi and Greedy for deploying user requests to satellite nodes.

Considering that the VNF placement strategies obtained by the proposed PGRA algorithm for user requests are sub-optimal VNF placement strategies, therefore, Gurobi Optimizer with the branch-and-cut algorithm in default configuration parameters [43] is used to find the global optimal VNF placement strategies for the maximum network payoff problem in equation (18) and then we evaluate the quality of the proposed PGRA algorithm. Under non-critical load conditions, the experiments with PGRA and Gurobi are conducted for $M = \{6, 8, 10, 12, 14, 16\}$ in satellite networks with $N = 6$ and 9, respectively. All the experiments run for 10 times and we obtain the average results, as shown in Table IX. We can observe from Table IX that the performance of Gurobi is better than that of the proposed PGRA algorithm in solving the VNF placement problem, however, the performance of the proposed PGRA algorithm is also guaranteed in finding sub-optimal VNF placement strategies. On average, the price of anarchy for all user requests is 98.62% in a satellite network with $N = 6$ and 99.85% in a satellite network with $N = 9$, where the price of anarchy is considered as the ratio of the network payoff of the proposed PGRA algorithm to that of Gurobi [46].

D. Performance Analysis in On-line Strategy

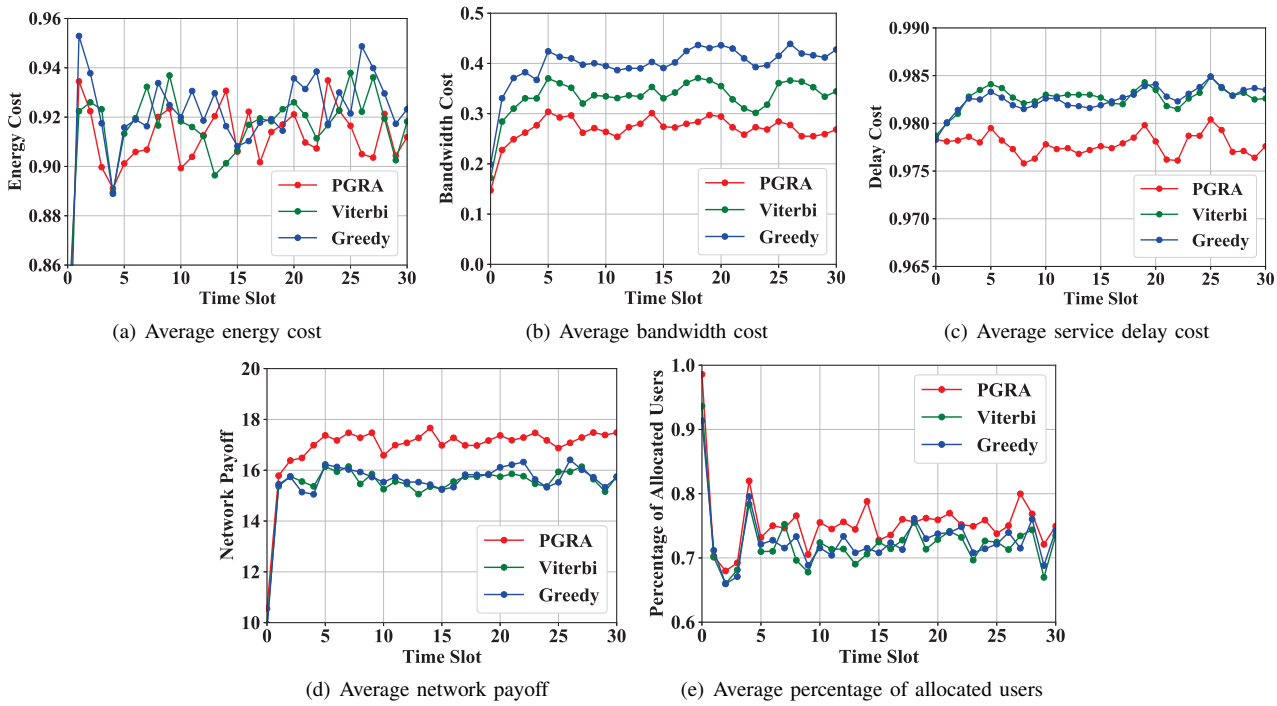
To evaluate the effectiveness of the proposed PGRA algorithm for addressing the VNF placement problem in dynamic environment, we make the following experiments in satellite networks with 6, 9, 12, and 15 satellite nodes, respectively. The total number of time slots for each instance is 50 and we randomly generate the number of user requests from 10 to 15 in each time slot. The running periods for user requests can be randomly selected from 1 to 4 time slots. When the running time for a user request is over, the resources used by the user request can free and be available for deploying new user requests in the next time slot. All experiments are performed for 10 times and we obtain the average results.

In the case of a satellite network with $N = 6$, the average results for different time slots in dynamic environment are described in Fig. 7. Fig. 7(a) illustrates the energy costs for different time slots. We can find that the energy costs for PGRA, Viterbi, and Greedy are comparatively close with each other. However, the proposed PGRA algorithm can reduce the average energy cost by 0.88% for Viterbi and 1.34% for Greedy. The average bandwidth costs for different time slots are described in Fig. 7(b), where we can observe that the proposed PGRA algorithm outperforms Viterbi and Greedy, and the average performance improvement of the proposed PGRA algorithm is 18.98% for Viterbi and 33.01% for Greedy. This is due to the fact that the lower is the bandwidth cost for a user request in a resource constrained scenario, the higher is the user payoff, and then the greater is the winning chance. Therefore, these user requests with minimum bandwidth costs will be deployed to satellites. The average service end-to-end delay costs are shown in Fig. 7(c). Similar to the results in Fig. 7(b), the delay costs of the proposed PGRA algorithm are better than that of Viterbi and Greedy, where the proposed PGRA algorithm improves the delay cost by 0.48% for the two baselines of Greedy and Viterbi. In addition, Fig. 7(d) shows the overall network payoffs for different time slots. We can find from Fig. 7(d) that the proposed PGRA algorithm performs better than Viterbi and Greedy for the network payoffs due to that an approximate VNF placement strategy is found by competing available resources with each other for maximizing user payoffs in each iteration procedure. On average, the proposed PGRA algorithm can increase the network payoff by 9.20% for Viterbi and 8.98% for Greedy, respectively. The percentages of allocated user requests for different time slots are shown in Fig. 7(e). We can also find from Fig. 7(e) that the proposed PGRA algorithm outperforms Viterbi and Greedy, where the proposed PGRA algorithm increases the percentage of allocated user requests by 4.25% for Viterbi and 3.94% for Greedy, respectively.

We also provide the average simulation results for different satellite networks with $N = 6$, 9, 12, and 15 in dynamic environment, which are described in Table X. We can find from Table X that the proposed PGRA algorithm outperforms the two baselines of Viterbi and Greedy in all cases. For the case of $N = 12$, the bandwidth cost obtained by the proposed PGRA algorithm reduces by 19.86% for Viterbi and 29.84% for Greedy, the delay cost obtained by the proposed PGRA algorithm reduces by 0.55% for Viterbi and 0.51% for Greedy. The proposed PGRA algorithm also improves the network payoff by 1.62% for Viterbi and 3.02% for Greedy, and increases the percentage of allocated user requests by 0.57% for Viterbi and 1.10% for Greedy, respectively. Note that the

TABLE X
PERFORMANCE COMPARISON FOR DIFFERENT SATELLITE NETWORKS WITH N = 6, 9, 12, AND 15 IN DYNAMIC ENVIRONMENT.

Name	Energy Cost			Bandwidth Cost			Delay Cost			Network Payoff			Percentage of Allocated Users		
	PGRA	Viterbi	Greedy	PGRA	Viterbi	Greedy	PGRA	Viterbi	Greedy	PGRA	Viterbi	Greedy	PGRA	Viterbi	Greedy
N=6	0.9101	0.9183	0.9225	0.2715	0.3352	0.4054	0.9782	0.9829	0.9829	16.999	15.5662	15.5982	0.754	0.7232	0.7254
N=9	0.9144	0.9228	0.9204	0.3282	0.4235	0.4965	0.9633	0.9704	0.9708	24.5116	23.349	23.2413	0.8869	0.8721	0.8706
N=12	0.8878	0.8948	0.8855	0.3465	0.4324	0.4939	0.9457	0.951	0.9506	30.4144	29.9268	29.5213	0.9598	0.9543	0.9493
N=15	0.8107	0.8144	0.7995	0.3258	0.374	0.4344	0.9257	0.9272	0.9274	33.8497	33.6261	32.9344	0.9936	0.9918	0.9852



(a) Average energy cost (b) Average bandwidth cost (c) Average service delay cost

(d) Average network payoff (e) Average percentage of allocated users

Fig. 7. Performance comparison for different time slots in a satellite network with 6 satellite nodes.

energy cost of the proposed PGRA algorithm is greater than that of Greedy. That is due to the fact that the proposed PGRA algorithm performs better than Greedy in resource allocation, then more user requests will be deployed to satellites and thus the energy consumption will increase accordingly. We can also find that when the number of satellites increases more available resources can be provided for user requests, which results in the increase in the network payoffs and the percentages of allocated user requests. For the proposed PGRA algorithm in dynamic environment, the network payoffs for N = 6, 9, 12, and 15 are 16.999, 24.5116, 30.4144, and 33.8497, the percentages of allocated user requests for N = 6, 9, 12, and 15 are 75.4%, 88.69%, 95.98%, and 99.18%, respectively. From Fig. 7 and Table X, we can demonstrate that the proposed PGRA algorithm is effective and efficient for addressing the VNF placement problem in dynamic environment.

VII. CONCLUSION

In this paper, we study the VNF placement problem by a potential game in satellite edge computing. We prove the VNF placement problem to be NP-hard and formulate the problem as a potential game with maximum network payoff. Each user request can make the deployment strategy in a self-interested way and all user requests can optimize their strategies by competing with each other in a distributed manner. Considering that a potential game admits at least a Nash equilibrium we implement a decentralized resource allocation algorithm based on a potential game to find approximate VNF placement strategies for user requests.

To evaluate the effectiveness of the proposed PGRA algorithm, we first discuss the influence of system parameters on the proposed PGRA algorithm performance by the Taguchi method. Then we make the experiments for different number of user requests in satellite networks with 6, 9, 12, and 15 satellite nodes and compare the simulation results with two baseline algorithms of Viterbi and Greedy. For example, in the case of N = 12, the average network payoff obtained by the proposed PGRA algorithm increases by 1.05% for Viterbi and 1.76% for Greedy, the average percentage of allocated user requests obtained by the proposed PGRA algorithm improves by 0.50% for Viterbi and 1.25% for Greedy. Based on the simulation results of Gurobi and PGRA, the average price of anarchy for all user requests can be 98.62% and 99.85% in satellite networks with N = 6 and 9, respectively. In dynamic environment, for N = 12, the proposed PGRA algorithm improves by 1.62% for Viterbi and 3.02% for Greedy in terms of network payoff, the percentage of allocated user requests obtained by the proposed PGRA algorithm increases by 0.57%

for Viterbi and 1.10% for Greedy. All the simulation results show that the proposed PGRA algorithm is an effective approach for addressing the VNF placement problem in satellite edge computing and performs better than Viterbi and Greedy.

REFERENCES

[1] M. De Sanctis, E. Cianca, G. Araniti *et al.*, "Satellite communications supporting internet of remote things," *IEEE Internet Things J.*, vol. 3, no. 1, pp. 113–123, 2016.

[2] N. Abbas, Y. Zhang, A. Taherkordi *et al.*, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, 2018.

[3] R. Xie, Q. Tang, Q. Wang *et al.*, "Satellite-terrestrial integrated edge computing networks: Architecture, challenges, and open issues," *IEEE Netw.*, vol. 34, no. 3, pp. 224–231, 2020.

[4] H. Huang, S. Guo, W. Liang *et al.*, "Green data-collection from geo-distributed IoT networks through low-earth-orbit satellites," *IEEE Trans. Green Commun. Netw.*, vol. 3, no. 3, pp. 806–816, 2019.

[5] Z. Zhang, W. Zhang, and F. Tseng, "Satellite mobile edge computing: Improving QoS of high-speed satellite-terrestrial networks using edge computing techniques," *IEEE Netw.*, vol. 33, no. 1, pp. 70–76, 2019.

[6] Z. Qu, G. Zhang, H. Cao *et al.*, "LEO satellite constellation for internet of things," *IEEE Access*, vol. 5, pp. 18 391–18 401, 2017.

[7] J. Wei and S. Cao, "Application of edge intelligent computing in satellite internet of things," in *Proc. IEEE Int. Conf. Smart Internet Things*, Tianjin, China, Aug. 2019, pp. 85–91.

[8] J. Gil Herrera and J. F. Botero, "Resource allocation in NFV: A comprehensive survey," *IEEE Trans. Netw. Serv. Manag.*, vol. 13, no. 3, pp. 518–532, 2016.

[9] G. Wang, S. Zhou, S. Zhang *et al.*, "SFC-based service provisioning for reconfigurable space-air-ground integrated networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 7, pp. 1478–1489, 2020.

[10] Y. Wang, J. Yang *et al.*, "Satellite edge computing for the internet of things in aerospace," *Sensors*, vol. 19, no. 20, pp. 4375–4380, 2019.

[11] N. Cheng, F. Lyu, W. Quan *et al.*, "Spaceaerial-assisted computing offloading for IoT applications: A learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1117–1129, 2019.

[12] S. Yang, F. Li, S. Trajanovski *et al.*, "Delay-aware virtual network function placement and routing in edge clouds," *IEEE Trans. Mobile Comput.*, vol. 20, no. 2, pp. 445–459, 2021.

[13] Y. Ma, W. Liang, M. Huang *et al.*, "Virtual network function service provisioning in MEC via trading off the usages between computing and communication resources," *IEEE Trans. Cloud Comput.*, 2020, doi:10.1109/TCC.2020.3043313.

[14] S. Song, C. Lee, H. Cho *et al.*, "Clustered virtualized network functions resource allocation based on context-aware grouping in 5G edge networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 5, pp. 1072–1083, 2020.

[15] Y.-Y. Shih, H.-P. Lin, A.-C. Pang *et al.*, "An NFV-Based service framework for IoT applications in edge computing environments," *IEEE Trans. Netw. Serv. Manag.*, vol. 16, no. 4, pp. 1419–1434, 2019.

[16] Z. Jia, M. Sheng, J. Li *et al.*, "VNF-based service provision in software defined LEO satellite networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 9, pp. 6139–6153, 2021.

[17] Z. Jia, M. Sheng, J. Li *et al.*, "Joint optimization of VNF deployment and routing in software defined satellite networks," in *Proc. IEEE Veh. Technol. Conf.*, Chicago, USA, Aug. 2018, pp. 1–5.

[18] Y. Cai, Y. Wang, X. Zhong *et al.*, "An approach to deploy service function chains in satellite networks," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp.*, Taipei, Taiwan, Jul. 2018, pp. 1–7.

[19] Q. He, G. Cui, X. Zhang *et al.*, "A game-theoretical approach for user allocation in edge computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 3, pp. 515–529, 2020.

[20] D. Monderer and L. Shapley, "Potential games," *Games Econ. Behav.*, vol. 14, pp. 124–143, May 1996.

[21] A. Moragrega, P. Closas, and C. Ibars, "Potential game for energy-efficient RSS-based positioning in wireless sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 7, pp. 1394–1406, 2015.

[22] F. Bari, S. R. Chowdhury, R. Ahmed *et al.*, "Orchestrating virtualized network functions," *IEEE Trans. Netw. Serv. Manag.*, vol. 13, no. 4, pp. 725–739, 2016.

[23] L. Yang, H. Zhang, X. Li *et al.*, "A distributed computation offloading strategy in small-cell networks integrated with mobile edge computing," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2762–2773, 2018.

[24] Y. Wang, J. Yang, X. Guo *et al.*, "A game-theoretic approach to computation offloading in satellite edge computing," *IEEE Access*, vol. 8, pp. 12 510–12 520, 2020.

[25] B. Denby and B. Lucia, "Orbital edge computing: Machine inference in space," *IEEE Comput. Archit. Lett.*, vol. 18, no. 1, pp. 59–62, 2019.

[26] F. Wang, D. Jiang, S. Qi *et al.*, "Fine-grained resource management for edge computing satellite networks," in *Proc. IEEE Global Commun. Conf.*, Waikoloa, USA, Dec. 2019, pp. 1–6.

[27] J. Wei, J. Han *et al.*, "Satellite IoT edge intelligent computing: A research on architecture," *Electronics*, vol. 8, no. 11, pp. 1247–1262, 2019.

[28] I. Leyva-Mayorga, B. Soret, and P. Popovski, "Inter-plane inter-satellite connectivity in dense LEO constellations," *IEEE Trans. Wireless Commun.*, vol. 20, no. 6, pp. 3430–3443, 2021.

[29] Q. Chen, G. Giambene, L. Yang *et al.*, "Analysis of inter-satellite link paths for LEO mega-constellation networks," *IEEE Trans. Veh. Technol.*, vol. 70, no. 3, pp. 2743–2755, 2021.

[30] M. Dayarathna, Y. Wen, and R. Fan, "Data center energy consumption modeling: A survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 732–794, 2016.

[31] T. Cao, C. Xu, J. Du *et al.*, "Reliable and efficient multimedia service optimization for edge computing-based 5G networks: Game theoretic approaches," *IEEE Trans. Netw. Serv. Manag.*, vol. 17, no. 3, pp. 1610–1625, 2020.

[32] H. Guo and J. Liu, "Collaborative computation offloading for multiaccess edge computing over fiber-wireless networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 5, pp. 4514–4526, 2018.

[33] R. Sridharan, "The capacitated plant location problem," *Eur. J. Oper. Res.*, vol. 87, no. 2, pp. 203–213, 1995.

[34] H. L. Choi and S. J. Lee, "A potential game approach for information-maximizing cooperative planning of sensor networks," *IEEE Trans. Control Syst. Technol.*, vol. 23, no. 6, pp. 2326–2335, 2015.

[35] J. Zeng, Q. Wang, J. Liu *et al.*, "A potential game approach to distributed operational optimization for microgrid energy management with renewable energy and demand response," *IEEE Trans. Ind. Electron.*, vol. 66, no. 6, pp. 4479–4489, 2019.

[36] M. Karimzadeh-Farshbafan, V. Shah-Mansouri, and D. Niyato, "Reliability aware service placement using a viterbi-based algorithm," *IEEE Trans. Netw. Serv. Manag.*, vol. 17, no. 1, pp. 622–636, 2020.

[37] Systems tool kit (STK). [Online]. Available: https://www.agi.com/products/stk

[38] STK's simplified general perturbations No. 4 (SGP4) propagator. [Online]. Available: https://pypi.org/project/sgp4

[39] A. Hagberg. Networkx: Network analysis in python. [Online]. Available: https://github.com/networkx

[40] Second quarter 2019 specpower_ssj2008 results. [Online]. Available: https://www.spec.org/power_ssj2008/results/res2019q2

[41] B. Kar, E. H.-K. Wu, and Y.-D. Lin, "Energy cost optimization in dynamic placement of virtualized network function chains," *IEEE Trans. Netw. Serv. Manag.*, vol. 15, no. 1, pp. 372–386, 2018.

[42] J. Liu, Y. Li, Y. Zhang *et al.*, "Improve service chaining performance with optimized middlebox placement," *IEEE Trans. Netw. Serv. Manag.*, vol. 10, no. 4, pp. 560–573, 2017.

[43] Gurobi optimizer reference manual. [Online]. Available: https://www.gurobi.com/documentation/9.1/refman/index.html

[44] S. Rao, P. Samant, A. Kadampatta *et al.*, "An overview of taguchi method: Evolution, concept and interdisciplinary applications," *Int. J. Sci. Eng. Res.*, vol. 4, no. 10, pp. 621–626, 2013.

[45] X. L. Zheng and L. Wang, "A multi-agent optimization algorithm for resource constrained project scheduling problem," *Expert Syst. Appl.*, vol. 42, no. 15-16, pp. 6039–6049, 2015.

[46] J. Zhang, S. Pourazarm, C. G. Cassandras *et al.*, "The price of anarchy in transportation networks: Data-driven evaluation and reduction strategies," *Proc. IEEE*, vol. 106, no. 4, pp. 538–553, 2018.

Xiangqiang Gao received the B.Sc. degree in school of electronic engineering from Xidian University and the M.Sc. degree from Xi'an Microelectronics Technology Institute, Xi'an, China, in 2012 and 2015, respectively. He is currently pursuing the Ph.D. degree with the School of Electronic and Information Engineering, Beihang University, Beijing, China. His research interests include rateless codes, software defined network and network function virtualization.

Rongke Liu received the B.S. and Ph.D. degrees from Beihang University in 1996 and 2002, respectively. He was a Visiting Professor with the Florida Institution of Technology, USA, in 2006; The University of Tokyo, Japan, in 2015; and the University of Edinburgh, U.K., in 2018, respectively. He is currently a Full Professor with the School of Electronic and Information Engineering, Beihang University. He received the support of the New Century Excellent Talents Program from the Minister of Education, China. He has attended many special programs, such as China Terrestrial Digital Broadcast Standard. He has published over 100 papers in international conferences and journals. He has been granted over 20 patents. His research interest covers wireless communication and space information network.

Aryan Kaushik is currently an Assistant Professor (Lecturer) at the University of Sussex, UK, with the School of Engineering and Informatics, from 2021. He has been a Research Fellow at the University College London, UK, with the Department of Electronic and Electrical Engineering, from 2020-21. He completed his PhD degree in Communications Engineering from The University of Edinburgh, UK, in 2019. He received his MSc degree in Telecommunications from The Hong Kong University of Science and Technology, Hong Kong, in 2015. He has held visiting research appointments at the Athena Research and Innovation Center, Greece, in 2021, the Imperial College London, UK, from 2019-20, the University of Luxembourg, Luxembourg, in 2018, and the Beihang University, China, from 2017-19. He has been a TPC Member for IEEE international conferences such as IEEE ICC 2021 and ICC 2022, and a Conference Champion for IEEE PIMRC 2020. His research interests include 5G6G wireless communications, signal processing, integrated communications and radar sensing, millimeter wave, multi-antenna communications and satellite communication networks.